|  |
| --- |
| EPAM Systems, RD Dep.  MTN.\*NIX.07 Oracle DB. Introduction to DWH |
| MTN.\*NIX.07 Labs - Access and Join Methods Part 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status of document | [**Kiryl Bucha**](mailto:Kiryl_bucha@epam.com) | 16-JAN-2012 |  |  |
|  |  |  |  |  |  |

*Contents*

[1. Auto Trace & Explain Plan 3](#_Toc315038644)

[1.1. Task 1: Auto Trace configuration training 3](#_Toc315038645)

[2. Join Methods 3](#_Toc315038646)

[2.1. Task 2: Nested Loops Joins 4](#_Toc315038647)

[2.2. Task 3: Sort-Merge Joins 4](#_Toc315038648)

[2.3. Task 4: Hash Joins 4](#_Toc315038649)

[2.4. Task 5: Cartesian Joins 4](#_Toc315038650)

[2.5. Task 6: Left/Right Outer Joins 4](#_Toc315038651)

[2.6. Task 7: Full Outer Join 4](#_Toc315038652)

[2.7. Task 8: Semi Joins 4](#_Toc315038653)

[2.8. Task 9: Anti Joins 5](#_Toc315038654)

[2.9. Task 10: Prepare summary table 5](#_Toc315038655)

# Auto Trace & Explain Plan

## Task 1: Auto Trace configuration training

Below all possible variants of SQL plus utilities autotrace:

set autotrace off

set autotrace on

set autotrace traceonly

set autotrace on explain

set autotrace on statistics

set autotrace on explain statistics

set autotrace traceonly explain

set autotrace traceonly statistics

set autotrace traceonly explain statistics

set autotrace off explain

set autotrace off statistics

set autotrace off explain statistics

**NOTE:** If you received next error: Check PLUSTRACE role is enabled. Please make next steps:

1. Run next script connected as sysdba:

# @ $ORACLE\_HOME/sqlplus/admin/plustrce.sql;

1. Grant role PLUSTRACE to $UserName$

# grant plustrace to $UserName$;

**Task Results:**

Expected:

Summary table with all result and text description of analyses this results.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Auto Trace Configuration Options | Expected Results | Description |
| 1 | off | Returned rows | does not display a trace report. |
| 2 | on | provides feedback including the returned rows, execution plan and statistics and the number of returned rows | displays a trace report |
| 3 | traceonly | provides feedback including the execution plan and statistics | displays a trace report, but does not print query data, if any |
| 4 | on explain | provides feedback including the returned rows, the number of returned rows and execution plan | shows the query execution path by performing an EXPLAIN PLAN |
| 5 | on statistics | provides feedback including the returned rows, the number of returned rows and statistics. | Displays statistics |
| 6 | on explain statistics | provides feedback including the returned rows, execution plan and statistics and the number of returned rows | displays a trace report and shows the query execution path by performing an EXPLAIN PLAN |
| 7 | traceonly explain | provides feedback including the execution plan | shows the query execution path by performing an EXPLAIN PLAN with no rows returned |
| 8 | traceonly statistics | provides feedback including the statistics and the number of returned rows | Displays statistics with no rows returned |
| 9 | traceonly explain statistics | provides feedback including the number of returned rows, execution plan and statistics | displays a trace report with no data rows |
| 10 | off explain | provides feedback including the returned rows, the number of returned rows | does not display a trace report. |
| 11 | off statistics | provides feedback including the returned rows, the number of returned rows | does not display a trace report. |
| 12 | off explain statistics | provides feedback including the returned rows, the number of returned rows | does not display a trace report. |

# Join Methods

**The Main Task** is to create SQL and prepare execution plan of statements with join methods on Task Topics (Task 2 - 9)

**Task Results:**

There are several tasks below with the same main expected result points:

* Create SQL using next tables: scott.emp, scott.dept
* Create additional needed Tables and Indexes
* Prepare screenshots of execution plan

## Task 2: Nested Loops Joins

**Example:**

# SELECT \*

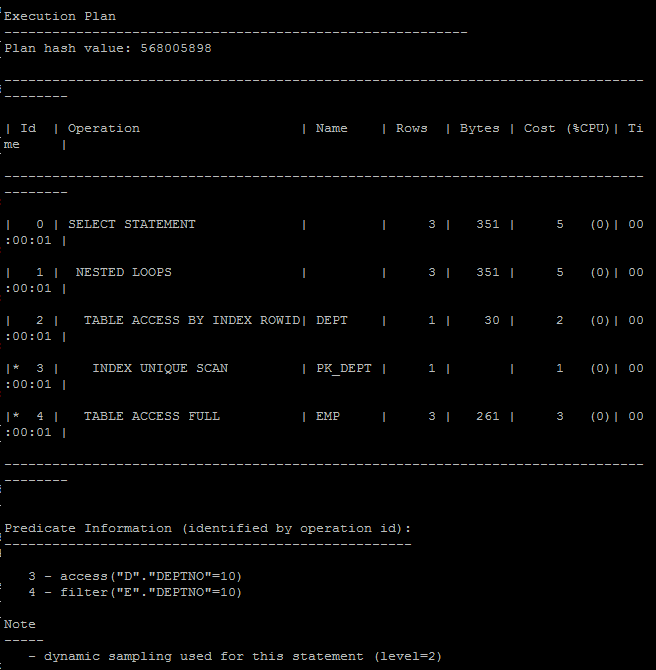
FROM scott.emp e, scott.dept d

WHERE e.deptno = d.deptno

AND d.deptno = 10

**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.

Set autotrace on explain;



**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_NL)

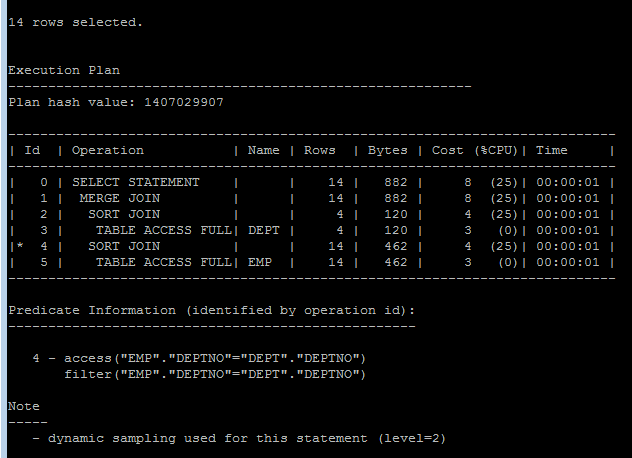
## Task 3: Sort-Merge Joins

**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.

SELECT /\*+ use\_merge (dept emp) \*/ empno, ename, dname, loc

FROM scott.emp, scott.dept

where emp.deptno = dept.deptno;



**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_MERGE)

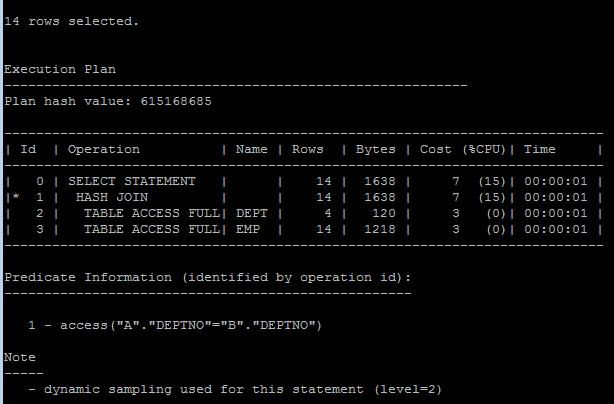
## Task 4: Hash Joins

**Task:** Prepare SQL **explain plan** using software: Oracle SQL Developer.

SELECT \*

FROM scott.emp a, scott.dept b

WHERE a.deptno = b.deptno



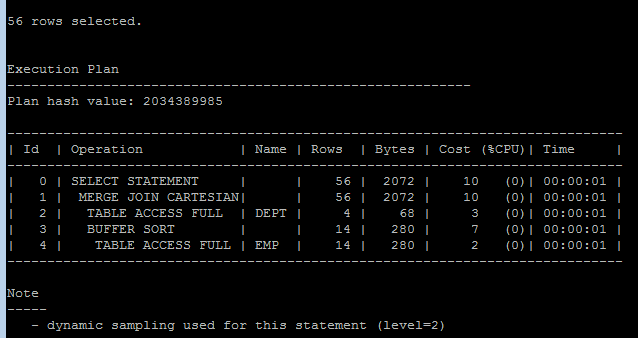
**Note:**  If you would like change in execution plan the type of join method use oracle performance hints. (USE\_HASH)

## Task 5: Cartesian Joins

**Task:** Prepare SQL **explain plan** using software: Oracle SQL Developer.

select empno, ename, dname, loc

from scott.emp, scott.dept



## Task 6: Left/Right Outer Joins

**Tasks:**

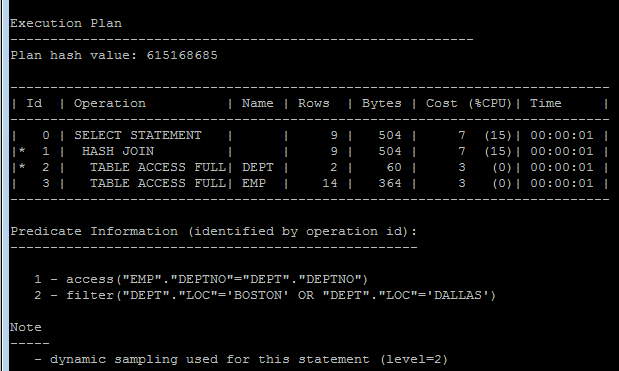
* Prepare SQL **trace protocol** using software: Oracle SQL Developer.
* Create SQL left outer join

select ename,job,dname

from scott.emp,scott.dept

where emp.deptno = dept.deptno and

loc in ('DALLAS','BOSTON')



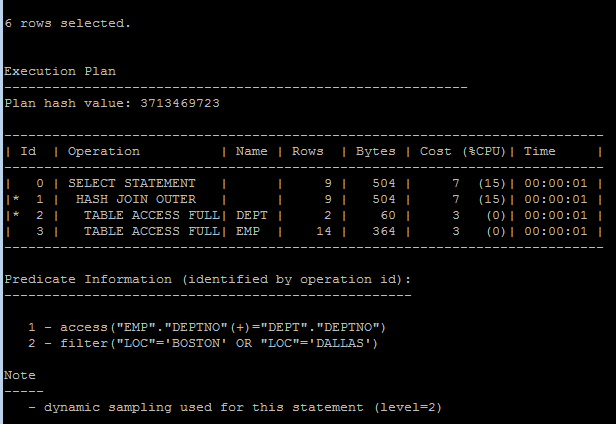
* Create SQL right outer join

select ename,job,dname

from scott.emp,scott.dept

where emp.deptno(+) = dept.deptno and

loc in ('DALLAS','BOSTON')



## Task 7: Full Outer Join

**Task:** Prepare SQL **explain plan** using: SQL plus Auto Trace Utility.

select ename,job,dname

from scott.emp,scott.dept

where emp.deptno = dept.deptno and

loc in ('DALLAS','BOSTON')

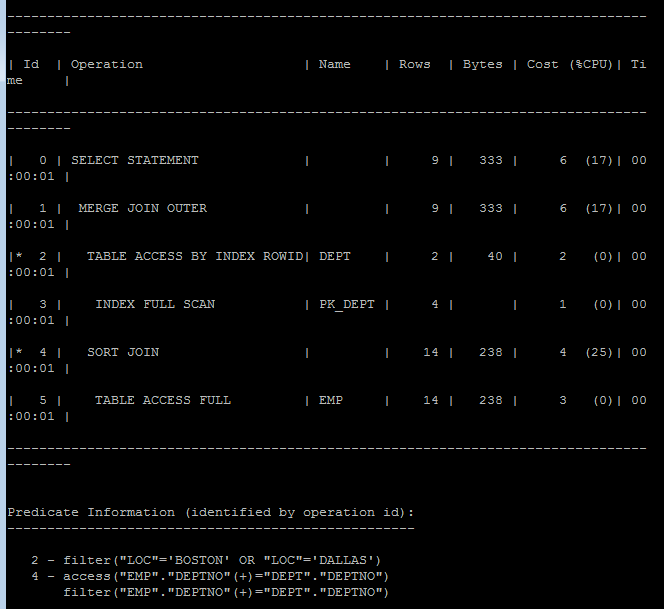
union

select ename,job,dname

from scott.emp,scott.dept

where emp.deptno(+) = dept.deptno and

loc in ('DALLAS','BOSTON')



## Task 8: Semi Joins

**Task:** Prepare All possible variants of SEMI JOIN SQL **explain plan** using: SQL plus Auto Trace Utility.

**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

1. SEMIJOIN – perform a semi-join (the optimizer gets to pick which kind)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE EXISTS

(

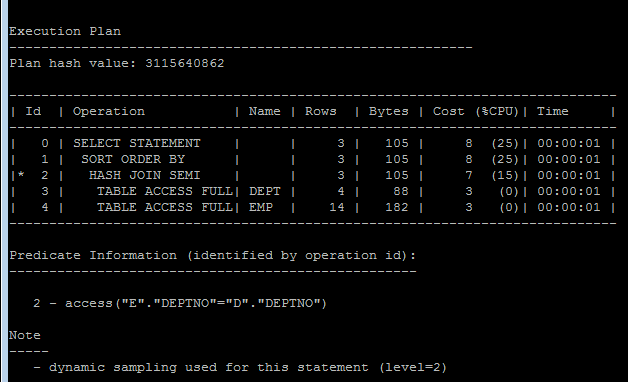
SELECT 1

FROM scott.emp E

WHERE E.deptno = D.deptno

)

ORDER BY D.deptno;



1. NO\_SEMIJOIN – obviously means don’t perform a semi-join

select D.deptno, D.dname

from scott.dept D

WHERE EXISTS

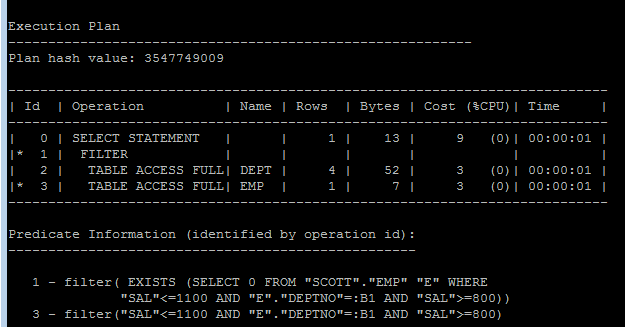
(

SELECT /\*+NO\_SEMIJOIN\*/ \*

FROM scott.emp E

WHERE E.deptno = D.deptno and sal between 800 and 1100

);



1. NL\_SJ – perform a nested loops semi-join (deprecated as of 10g)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE EXISTS

(

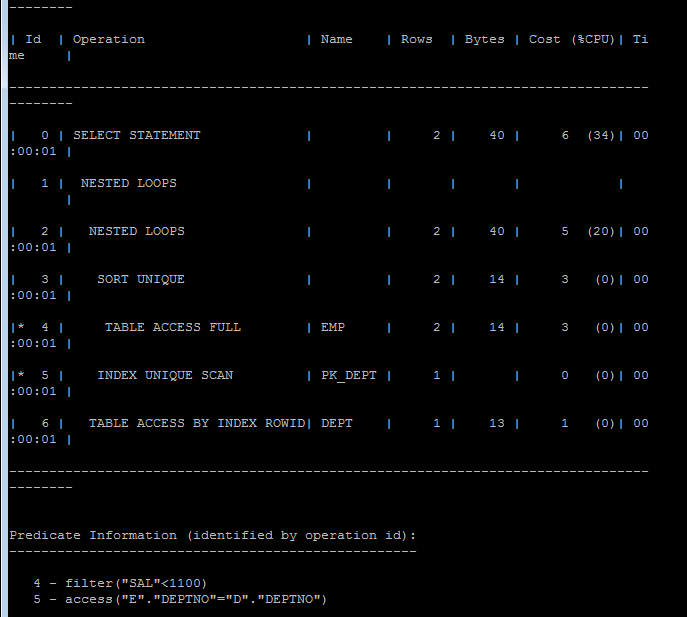
SELECT /\*+ NL\_SJ (dept emp) \*/\*

FROM scott.emp E

WHERE E.deptno = D.deptno and sal < 1100

)

ORDER BY D.deptno;



1. HASH\_SJ – perform a hash semi-join (deprecated as of 10g)

select D.deptno, D.dname

from scott.dept D

WHERE EXISTS

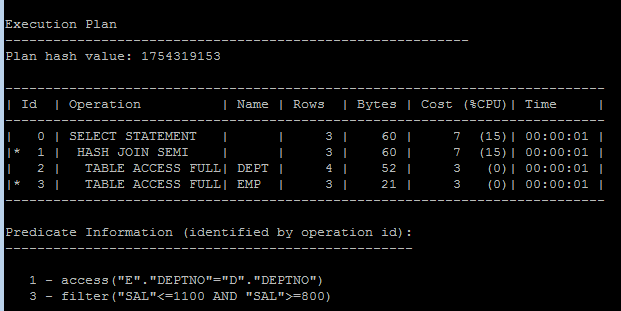
(

SELECT /\*+HASH\_SJ\*/ \*

FROM scott.emp E

WHERE E.deptno = D.deptno and sal between 800 and 1100

);



1. MERGE\_SJ – perform a merge semi-join (deprecated as of 10g)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE EXISTS

(

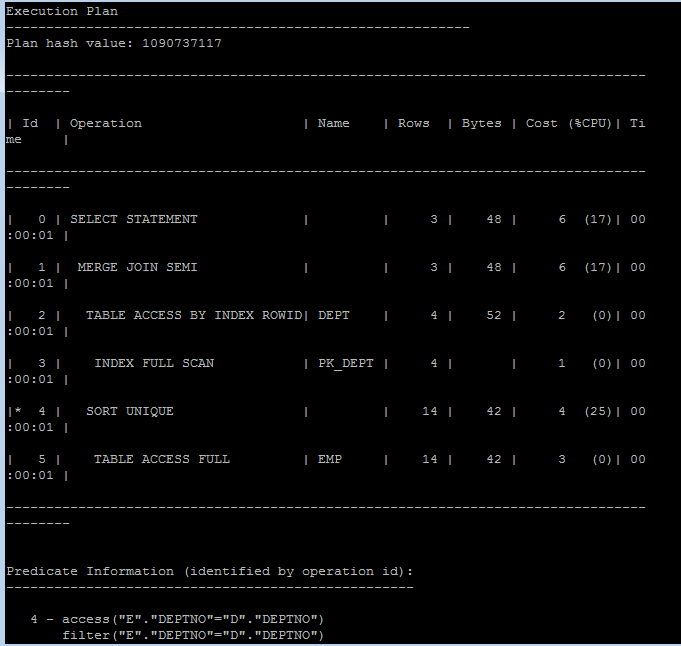
SELECT /\*+ MERGE\_SJ (dept emp) \*/1

FROM scott.emp E

WHERE E.deptno = D.deptno

)

ORDER BY D.deptno;



## Task 9: Anti Joins

**Task:** Prepare All possible variants of ANTI JOIN SQL **explain plan** using: SQL plus Auto Trace Utility.

**Note:**  If you would like change in execution plan the type of join method use oracle performance hints.

1. ANTIJOIN – perform an anti-join (the optimizer gets to pick which kind)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE NOT EXISTS

(

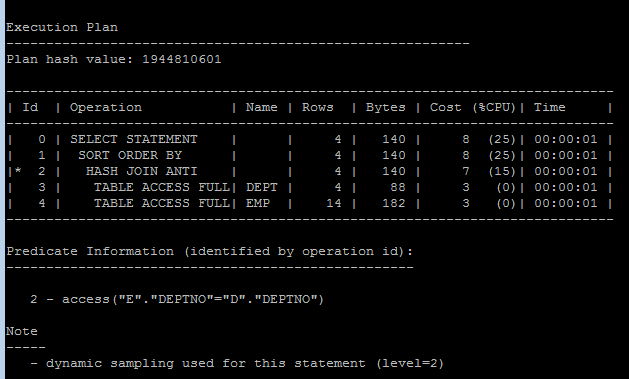
SELECT 1

FROM scott.emp E

WHERE E.deptno = D.deptno

)

ORDER BY D.deptno;



1. USE\_ANTI – older version of ANTIJOIN hint

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE NOT EXISTS

(

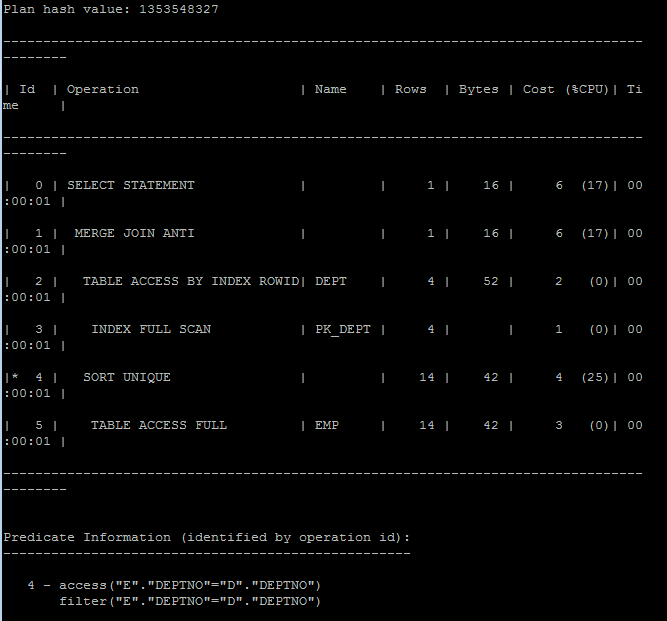
SELECT /\*+USE\_ANTI\*/ 1

FROM scott.emp E

WHERE E.deptno = D.deptno

)

ORDER BY D.deptno;



1. NL\_AJ – perform a NESTED LOOPS anti-join (deprecated as of 10g)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE NOT EXISTS

(

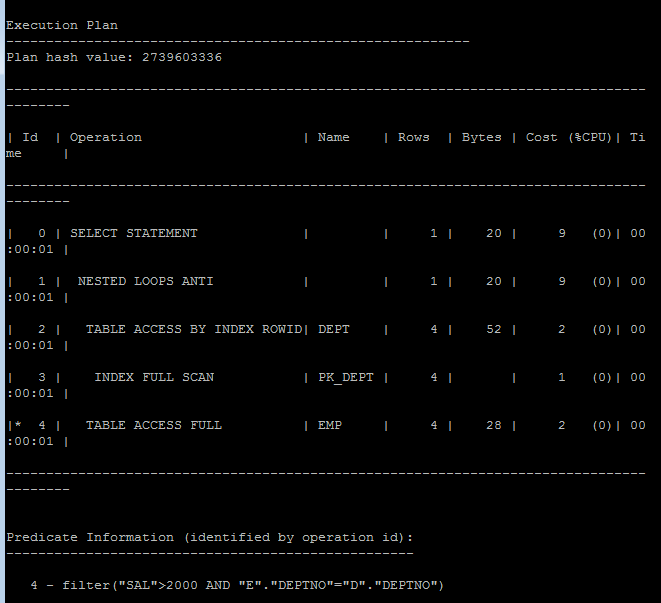
SELECT /\*+NL\_AJ\*/ 1

FROM scott.emp E

WHERE E.deptno = D.deptno and sal > 2000

)

ORDER BY D.deptno;



1. HASH\_AJ – perform a HASH anti-join (deprecated as of 10g)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE NOT EXISTS

(

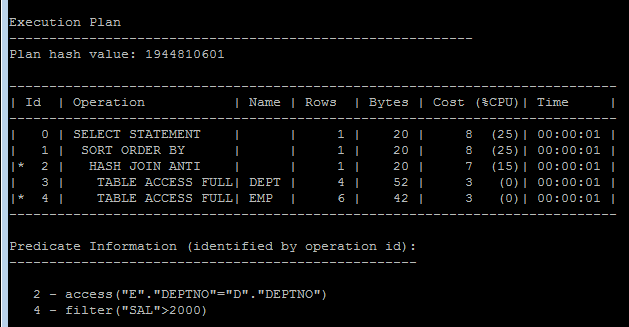
SELECT /\*+HASH\_AJ\*/ 1

FROM scott.emp E

WHERE E.deptno = D.deptno and sal > 2000

)

ORDER BY D.deptno;



1. MERGE\_AJ – perform a MERGE anti-join (deprecated as of 10g)

SELECT D.deptno, D.dname

FROM scott.dept D

WHERE NOT EXISTS

(

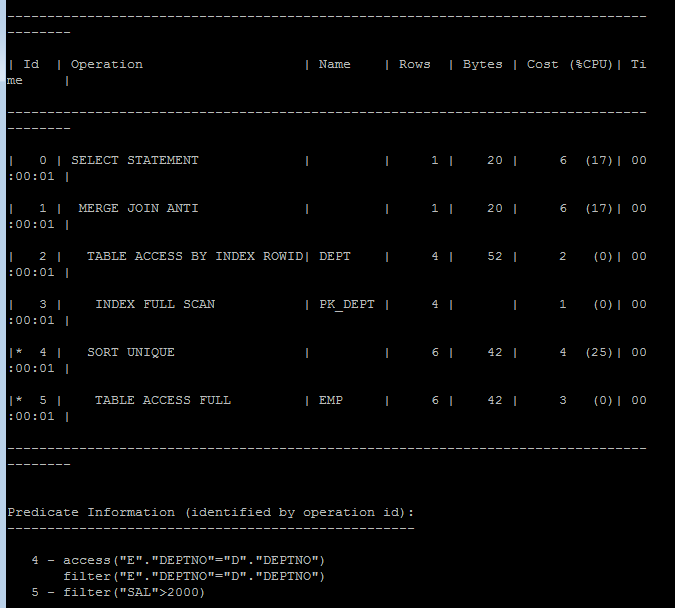
SELECT /\*+MERGE\_AJ\*/ 1

FROM scott.emp E

WHERE E.deptno = D.deptno and sal > 2000

)

ORDER BY D.deptno;



## Task 10: Prepare summary table

**Task:** Make comparison of all possible variant of join methods and join access methods and fill the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Join Access “A” | Join Access “B” | Nested Loop | Hash Join | Sort-Merge Join | Anti-Join | Semi-Join |
| Small Table | Small Table | The nested loop iterates over all rows of the *outer table*. If there are conditions in the where clause of the SQL statement that apply to the outer table only, it checks whether those apply. The rows from the inner table are either found using an index (if a suitable exists) or by doing a full table scan. So it has better performance in case we have small tables or outer small and indexed inner table. | A hash join (ideally) takes the smaller table and performs a hash algorithm on the columns for the where conditions between the tables and stores the result. After it has finished, it iterates over the other table and performes the same hashing algorithm on the joined columns. It then searches the previously built hashed values and if they match, it returns the row. So better performance (as we perform full access) will be for smaller tables, one smaller the the other and indexed ones. | A **merge join** basically sorts all relevant rows in the first table by the join key , and also sorts the relevant rows in the second table by the join key, and then merges these sorted rows. So it have high performance for small tables getting worth up to the increasing its size. | Anti-joins are basically the same as semi-joins in that they are an optimization option that can be applied to nested loop, hash, and merge joins. However, they are the opposite of semi-joins in terms of the data they return. Those mathematician types familiar with relational algebra would say that anti joins can be defined as the complement of semi-joins. | The main difference between a normal inner join and a semi-join is that with a semi-join, each record in the first set is returned only once, regardless of how many matches there are in the second set. This definition implies that the actual processing of the query can be optimized by stopping search in the 2nd table as soon as the first match is found. It is evident, that better performance will be for small tables or if one is much smaller then the other. Indexed tables will perform better result. |
| Small Table | Indexed Small Table |
| Small Table | Clustered index table |
| Small Table | Hash clustered table |
| Small Table | Unique indexed table |
| Small Table | Big Table |
| Big table | Big table |